LOGISTIC REGRESSION IN AN ADAPTIVE WEB CACHE

ANNIE P. FOONG, YU-HEN HU, AND DENNIS M. HEISEY
University of Wisconsin

Adaptive Web caching offers a way to improve network performance using current technologies. The challenge facing researchers is to determine what objects should be cached. As biostatisticians use regression models to predict the risk of disease development, the authors use a logistic regression model to predict the "cache worthiness" of Web objects.

The World Wide Web has become a victim of its own success. While Web usage has increased exponentially, the available network infrastructure has not. As a result, the network cannot keep up with user demands, and performance suffers. For example, a survey of text-only download of home pages of 20 popular sites (from 27 cities over a T1 connection) revealed download times ranging from 3.36 to 16.29 seconds.[1] The low end of such performance is obviously not acceptable.

Factors leading to the increase in Web traffic and subsequent low performance include

- the increasingly multimedia nature of Web content;
- the widespread use of push technology;
- a mass migration of traditional services to Web-based applications (for example, online banking and investing);
- millions of new users due to the high competition between Internet service providers and subsequent low-cost Internet access.

Researchers are exploring improvements to Internet performance from several angles:

- *Infrastructure.* Experimental gigabit networks are currently under construction. Cable modem and asymmetrical digital subscriber loop (ADSL) technologies will enable multimegabit access from homes.
- *Protocols.* The design of new hypertext transfer protocols (for example, HTTP-next generation) aims to reduce inherent inefficiencies of HTTP 1.0, which establishes a new TCP connection for every file request.
- *Compression.* Use of better encoding techniques, especially for multimedia files, can significantly reduce traffic.

 1089-7801/99/$10.00 ©1999 IEEE

- *Caching*. Caches placed at various points on the Web can dramatically reduce the need to repeatedly reload the same Web page from a remote site. A similar but resource-intensive approach is to mirror entire sites on several servers and thus bring content physically closer to the end user.

Experience has shown that no matter how much CPU power or communication bandwidth increases, demand will always quickly outpace supply.[2] Furthermore, proposed protocol optimizations do little to improve performance at current modem, ISDN, and LAN speeds.[3] JPEG and GIF are compressed image file formats (between one-tenth and one-fiftieth the original size) that make graphically rich Web pages possible.[4] However, compressing and decompressing files require extra processing time and intensive CPU cycles.

Of the four approaches, caching promises the greatest performance gain and can be implemented with current technologies. It is also the only approach that addresses the physical distances between users and Web objects.

Most of the popular Web browsers implement client caches. Accessed objects are copied on the users' hard disk, circumventing the need for an Internet connection the next time the object is accessed. By the same token, a server can cache high-demand objects on local disks, thereby reducing the need to transfer the objects from their home locations, which may be on other network drives.

Given a finite cache space, the challenge is to determine what objects are worthy of caching. To do so, we have designed an adaptive Web-caching strategy based on user re-access patterns. Our premise is that we can predict future Web accesses by learning an object's access history.

CACHING MOVES TO THE WEB

Traditional caches see a fixed-size page; a Web cache, on the other hand, sees complete objects (text files, images, or video clips), which vary considerably in size. In addition, a traditional cache deals with addresses, while a Web cache can potentially deduce more contextual information from its objects. Web objects are predominantly read-only, making implementation of cache coherence easier. The response times of Web accesses are in the order of seconds (versus milliseconds for system access), which allows for more elaborate caching algorithms. Finally, a Web cache encounters more dimensions of dependence than are taken into account by traditional methods.

Researchers have focused on two general approaches to Web caching: empirical observations and experiments, or implementation; and modeling.

Implementation

The least-recently-used (LRU) and least-frequently-used (LFU) replacement strategies are generally accepted for traditional system caches. However, there is no agreed-upon strategy for Web caches. Cao and Irani identified at least nine techniques for Web caching, including variations of LRU and LFU.[5] Most researchers add secondary factors such as size and last-accessed times to improve performance.[6,7] Cao and Irani's technique is based on the minimization of a user-determined cost.[5]

The existence of so many replacement algorithms is mainly due to the variable-sized objects a Web cache sees. While there are optimal (off line) replacement algorithms for uniform-sized cache objects, the determination in a variable-sized case is computationally intractable.[8] Replacement strategies are therefore deduced through observation and ad hoc heuristics.

Modeling

There have been numerous attempts to determine a universal model for Web traffic. These are usually based on statistics collected from various sites over extended periods. The primary goal of such modeling is to better understand the overall workings of the Web and to develop realistic synthetic Web workloads.

Lorenzetti, Rizzo, and Vicisano derived a re-access probability as a function of file size, number of past accesses, and time since last access.[9] The probability is obtained by manually fitting an exponential function to empirical data. Almeida and de Oliveira based their formulation on the distance between successive accesses to cached objects.[10] Arlitt and Williamson attempted to deduce universal truths about Web traffic by identifying invariants.[11] They suggested that LFU is preferable to LRU since it is common to find a high concentration of references (number of past accesses). Crovella and Bestavros based their model on transmission times.[12] Most researchers agree that long-term WWW accesses exhibit chaotic (self-similar) behavior.

Lorenzetti, Rizzo, and Vicisano assumed that the same probability distribution applies across the entire Web.[9] Their methodology is useful in determining probabilities of the fitted set of data, but it may be difficult to reproduce the model in different situations. Furthermore, descriptive statistics (such as maximums, means, and variances) give

typical values, which are not necessarily indicative of the Web accesses a particular cache may see. Such values may not directly affect caching. For example, Arlitt and Williamson showed that HTML and image files account for more than 90 percent of Web accesses.[11] However, caching only HTML or image files may not lead to good performance if they are not prone to re-access.

A formalized Web model should allow us to systematically arrive at replacement heuristics. Our goal is to develop an adaptive cache strategy to accommodate the heterogeneity of the Web. Moreover, our strategy must be able to model multiple factors adequately.

LEARNING FROM EXAMPLE

A logistic regression model is a simple and easily repeatable method of analysis. Our assumption is that Web access patterns over the short term are not completely random nor chaotic (self-similar). We are not only interested in a model's ability to fit current data but also in its ability to predict. In this way, we can directly use the model to guide our choice of cache strategies. Web pages deemed most likely to be re-accessed in the near future will be admitted and kept in cache. We pose this as a "learning from example" problem in machine learning theory: the *predicate* to be learned is Web object re-access; the *examples* are traces of past Web accesses; and a logistic regression model accomplishes the *learning*.

Re-access as the Event

Regression models have been widely used by biostatisticians to model the risk (or probability) of disease development (the event) as a function of factors suspected to affect the disease.[13] These factors are also referred to as *predictors* or *covariates* of the model. Our goal is to express the outcome of the dependent variable Y (of some value g), in terms of its predictors, $(1, X_1, \ldots, X_k)$ and their respective coefficients $(\beta_0, \beta_1, \ldots, \beta_k)$.

$$P(Y = g \mid 1, X_1, \ldots, X_k) = f(z)$$

where z takes on a linear form, such that

$$z = \sum_{j=0}^{k} \beta_j X_j$$

Given a set of observed data, the coefficients can be estimated by a suitable method (learning phase). Once the coefficients are determined, we can use the function to predict the outcome of an event when the predictors are known (prediction phase).

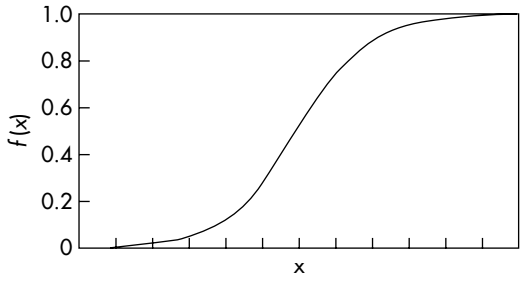

Figure 1. The logistic function, $f(x) = 1/(1 + e^{-x})$.

In the simple case, Y is a binary variable. It takes on the value of one or zero, according to whether or not a specific event (for example, heart disease) occurs during the study period (time t_0 to t). Thus the model allows us to determine the risk factors of disease. A sample population is surveyed. The result may show that people with high cholesterol levels are more prone to heart disease. The underlying assumption is that if a high cholesterol level is predictive of heart disease in the sample population, the same is true of the general population.

In our context, the event of interest is whether an object is re-accessed at least once in the next W_F accesses. Predictors include

- size and type of object,
- number of times it has been accessed, and
- time since last access.

We want to ascertain whether such factors are predictive of Web re-accesses.

The Logistic Function and the Web

Let $P(Y = 1|\mathbf{X})$ be the probability of the event Y = 1, given the predictor vector **X**. This is modeled as a function of linear predictor z, $P(Y = 1|\mathbf{X}) = f(z)$. Since $P(Y = 1|\mathbf{X})$ is a probability, the function f(z) must map the real line between 0 and 1. Several so-called link functions are commonly used for this purpose, the most popular being the logistic function (Figure 1) taking the form

$$P\left(Y = 1 \middle| \mathbf{X}\right) = \frac{1}{1 + \exp(-z)} \qquad -\infty < \mathbf{z} < +\infty \tag{1}$$

The logistic function has several favorable properties:

- it maps probabilities to 0 and 1;

- it lends itself to biologically meaningful interpretation;[13]
- its "S" shape indicates that the effect of z is minimal for low zs until some threshold is reached.

When enough predictors are present, the function rises rapidly and remains relatively constant once z gets large. Part of our investigation is to determine whether such a function can be applied to the Web. This specific model is referred to as the logistic regression model.

A nonevent is simply modeled as the complement of an event, or

$$P(Y = 0|\mathbf{X}) = 1 - P(Y = 1|\mathbf{X}).$$

Suppose Y_i is the binary outcome (1 for an event, 0 for a nonevent) observed on the ith access. Let $\mathbf{X}_i$ be the associated vector of predictors. Assuming $Y_1, Y_2, ..., Y_m$ are independent, the joint probability of the sequence $Y_1, Y_2, ..., Y_m$ is proportional to the product

$$\prod_{i=1}^{m} P\left(Y = Y_i \middle| \mathbf{X}_i\right)$$

This is the likelihood equation. The optimal coefficients β are found by maximizing the likelihood equation, resulting in maximum likelihood estimates. Iterative algorithms, such as Newton-Raphson, are usually employed to perform the maximization.

The coefficient of a predictor variable indicates how it influences the probability of an event. If $\beta = 0$, the variable has no influence. If $\beta > 0$, the variable increases the probability of the event, and if $\beta < 0$, the variable decreases the probability of the event. However, it must be noted that the value of β alone does not govern the magnitude of influence. It is the product of the coefficient and predictor ($\beta\mathbf{X}$) that affects the outcome. Only estimates of the β coefficients are obtained. The p-value is commonly used to measure the strength of evidence that the estimates of β are statistically significantly different from zero.[13] For our purposes, we consider any predictors with p-value ≤ 0.1 to be significant.

CAPTURING PATTERNS

We chose to work with application-level statistics, which are currently available as log files, and work within existing protocols (HTTP 1.0 and 1.1). We work primarily with server logs as they are readily available and do not involve the privacy concerns of proxy and client logs. As such, we are capturing the re-access patterns of a server cache and dealing with objects not served by downstream client and proxy caches. We have also restricted our predictors to those easily available from log files.

The problem description becomes

Y = 1 document re-accessed in a given forward-looking window, W_F
= 0 otherwise
X_1 = size of document
X_2 = type of document (categorical variable)
1: HTML and text files (html, shtml, txt)
2: special format files (ps, pdf, rtf)
3: image files (jpeg, giff)
4: audio files (aud, snd, wav)
5: video files (avi, qt, mpeg)
6: executables (pl, exe)
0: all others
X_3 = number of previous hits (within a backward-looking window, W_B)
X_4 = time since last access

Cache designs are typically validated by accesses produced by actual workloads (also known as trace-driven simulation).[14] We used traces (available at the Internet Traffic Archive[15]) from five Web servers:

- one day's requests to the EPA server at Research Triangle Park, North Carolina (EPA);
- one year's requests to the University of Calgary Computer Science Department server, Canada (CAL);
- two months' requests to the NASA Kennedy Space Center server, Florida (NASA);
- seven months' requests to the University of Saskatchewan server, Canada (SASK); and
- two weeks' requests to ClarkNet, a commercial ISP server, Washington, D.C. (CLAR).

The traces contain information about requesting host, time of request, requested URL, number of bytes in reply (file size), and reply status. We preprocessed the traces to remove badly formed lines and extraneous characters. Files with the same URL but of different size were considered modified versions. When accessing a file from a server where caching is enabled, a "Not modified 304" status may be returned, indicating that the file is served directly from the cache. Because these files contain no size information, we use the last known size from its trace history. Files without extensions are considered to be of type 1 (HTML/text) objects.

The major drawback in our strategy is its added complexity. However, the logistic regression model is a relatively inexpensive process. The learning

Table 1. Trace characteristics and significant predictors of the LR model.

Trace	Total size used in prediction (MB)[3]	Minimum size needed for infinite cache (MB)	Coefficients associated with predictors	Standard error	p-value[1]
EPA	39.7	16	0.58 x BHITS	0.13	0.0001
			–7.2e–7 x SINCE	1.8e–7	0.0001
NASA	39.9	32	0.78 x BHITS	0.16	0.0001
			(–, 17.5, -5.7, 18.4, –, –, –) x TYPE	–[2]	0.02
			–4e-5 x SIZE	1.5e-5	0.06
			–1.4e-5 x SINCE	4.2e-6	0.0007
SASK	31.9	3.2	0.45 x BHITS	0.12	0.0003
			–7.8e-6 x SINCE	3.1e–6	0.01
CLAR	46.5	18	–1.29 x BHITS	0.80	0.1
			–6e–6 x SINCE	1.4e–6	0.0001
CAL	46.3	7	0.66 x BHITS	0.13	0.0001
			–9.3e-7 x SINCE	1.9e–7	0.0001

[1] The p-value can be obtained using the Wald statistic.[13]

[2] Standard errors are not available for polytomous predictors.

[3] Total number of bytes involved in prediction phase.

phase (which incurs the highest cost) took between 1.3 to 1.5 seconds of processing using a commercial package (SAS) on a Pentium 133-MHz system. Customized code should reduce processing time. With Web access times in the seconds range, additional processing should not be a bottleneck. We have yet to determine the model's implementation and feasibility in real time.

Our caching algorithm can theoretically be implemented as two background processes: continuous learning and predicting. The cache strategy itself (prediction phase) requires minimal overhead: the computation of a probability given in Equation 1.

For simplicity, we performed only one pass of learning and prediction. Learning was done on N_L accesses and prediction on the next N_P accesses (Figure 2a). Because we based our learning on the number of accesses, or events, instead of time, we were able to circumvent the possibility of down-time on the servers. Furthermore, we ensured that each server had equal learning. We also specified the windows, W_F and W_B, in terms of events. N_L events were submitted as the learning data to obtain the maximum likelihood estimates of the coefficients. The coefficients for the five traces are given in Table 1.

We then use the estimated coefficients to calculate the probability of re-access for any given Web object in the next N_P accesses using Equation 1. We have arbitrarily chosen the following values for our experiments: $N_L = 500$, $N_P = 5{,}000$, $W_F = 50$, $W_B = 50$.

For example, the NASA server log has the following characteristics:

$$\hat{\beta} = \begin{bmatrix} -18.28 \\ 0.78 \\ (-, 17.5, -5.7, 18.4, -, -, -) \\ -4.0e-5 \\ -1.4e-5 \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} 1 \\ BHITS \\ TYPE \\ SIZE \\ SINCE \end{bmatrix}$$

Applying the coefficients to Equation 1, an object with BHITS=0, TYPE=1 (HTML/text), SIZE= 1,000 bytes, SINCE=100 seconds, has

$$\mathbf{P}(\text{re-access}) = 1/(1+\exp(-18.28 + 0 + 17.5 - 4.0e\text{–}5*1000 - 1.4e\text{–}5*100)) = 0.61$$

An object with TYPE=2 (special format) with similar characteristics has

$$\mathbf{P}(\text{re-access}) = 1/(1 + \exp(-18.28 + 0 - 5.7 - 4.0e\text{–}5*1000 - 1.4e\text{–}5*100)) \approx 0$$

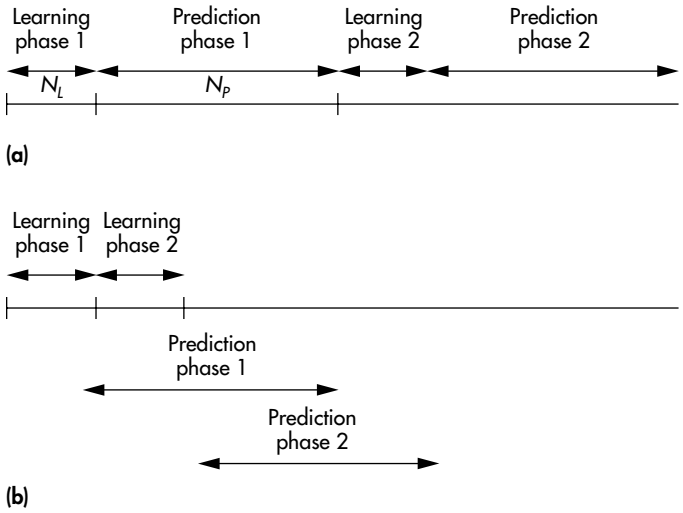

Figure 2. Learning and predictive phases: (a) distinct, (b) continuous.

Given these odds, the first object will be cached in favor of the second. The observation is that special-format files (such as PostScript and RTF) on the NASA server are less likely to be re-accessed. Such a distinction is not characteristic of all servers.

MEASURING DELAY

The ultimate goal of a cache system is to reduce the access times for the end user. The classical textbook formula of average access time of an object in system cache is as follows:

Average access time = $t_{cache} + (1 - h) * t_{noncache}$ [14]

t_{cache} = Time needed to transfer object from cache

$t_{noncache}$ = Time needed to transfer object from noncache location

h = Object hit ratio

This formula assumes that transfer times are independent of object size, network conditions (that is, the availability of bandwidth), and object location. These assumptions are valid for system caches since they deal with fixed-size pages; stable memory/disk connection; and objects residing at similar distances from point of processing.

When an object is not in the cache, the Web server retrieves it from its local disk or a local area networked disk. Since we are dealing with a LAN, assumptions about network condition and object location hold. However, the assumption that Web objects will be a fixed size is not true. As a result, a better performance metric relating to access time is the byte hit ratio, such that

Average access time per byte = $t_{cache} + (1 - b) * t_{noncache}$

Average access time of object i = $(t_{cache} + (1 - b) * t_{noncache}) *$ size of object I

t_{cache} = Time needed to transfer 1 byte from cache

$t_{noncache}$ = Time needed to transfer 1 byte from noncache location

b = Byte hit ratio

We identified the following to be useful:

$$\text{Byte hit ratio} = \frac{\text{Number of bytes served from cache}}{\text{Total number of bytes}}$$

$$\text{Object hit ratio} = \frac{\text{Number of objects served from cache}}{\text{Total number of objects}}$$

$$\text{Object replacement ratio} = \frac{\text{Number of objects replaced from cache}}{\text{Total number of objects}}$$

The byte hit ratio also gives us a direct measure of raw bandwidth requirements. Where Internet usage is charged by volume, this metric relates to cost. The object hit ratio also gives us a measure of what we term *user-perceived response time*. A typical Web page contains 10–50 independent Web objects. The ability to serve at least one of these objects while the rest of the Web page loads will improve apparent delay time for the user. A high object hit ratio also reduces the number of requests a server must process.[11]

Finally, we introduce an object replacement ratio. Reducing the replacement ratio reduces disk fragmentation and the need for constant cache maintenance. One-time referencing behavior can result in the cluttering of one-third of a server cache with useless files.[11] A low replacement ratio, therefore, also indicates caching efficiency. Other estimates of cache performance use similar metrics since they can be easily computed with an off line simulator. These static metrics are relatively representative of server (and system) cache performance. However, assumptions about network condition and object location do not hold for proxy and client caches. Thus we must dynamically determine cache performance using an online simulator; hit ratios alone will not be sufficient.

We determined the minimum cache size needed for emulating an infinite cache for each of the traces and ran simulation on cache sizes that were 1, 5, 10, and 20 percent of the infinite cache size. This ensured that each trace had ample chance to "warm" the cache and that comparisons were standardized across the different traces. The infinite cache size may seem small since we are looking to predict for 5,000 events. We also determined the hit ratios for an infinite cache and present results relative to this ideal.

Table 2. Average performance scores (best: 5, worst: 1).

Policy	Object hit ratios	Byte hit ratios	Object replacements
LR	4.15	4.25	4.95
LFU	4.4	3.25	3.85
LRU	3.1	3.9	2.9
FIFO	2.15	2.35	2.5
Random	1.25	1.25	1.15

COMPARING PERFORMANCE

We compared the performance of our logistic-regression-based policy to the more traditional policies—LRU, FIFO, LFU, and random. In the LR policy, we replace objects with the lowest re-access probability value. If an arriving object is found to have a probability lower than those already in the cache, the object is not cached at all. Williams and Abrams et al. have indicated that when size limitation is added to any of the policies, object hit ratio is improved.[6,7] However, the byte hit ratio will suffer. As our objective is to determine the viability of the LR model, we limit our comparative study to the generic version of each policy.

To allow for easier comparison, we assigned a score (best: 5, worst: 1) to how each policy ranked for every trace, taken over different cache sizes. For example, in the EPA trace for object hit ratios at the 1 percent relative cache size (Figure 3), the LFU strategy receives a score of 5; LR, 4; LRU, 3; FIFO, 2; and random, 1. The average scores are summarized in Table 2. Our LR policy is the best performer for byte hit ratios (Figure 4) and object replacement ratio (Figure 5). It is the second best performer in terms of object hit ratios (Figure 3).

The resulting model is intuitively appealing. "Number of hits in W_B" and "time since last access" are found to be predictors of future re-accesses. The model indicates that more hits in W_B will result in a higher probability of re-access (basis of LFU). It also shows that the longer the "time since last access," the lower the probability of re-access (the basis of LRU).

As such, we expect the performance of LR policy to be similar to the LFU and LRU policies in traces where these predictors are significant. LR's learning window is fixed at 50 past events. As cache sizes increase, both LFU and LRU learn more from the cache history than the LR algorithm. We acknowledge this oversight, and future work will ensure that all algorithms have equal learning opportunities.

It is important to note the two traces (CLAR and NASA) where our LR algorithm works extremely well. In the CLAR trace, the model indicates that more hits in the past actually leads to lower probability of re-access. In this trace, LR performs much better than LFU. In the NASA trace, all four predictors are significant. Once again, LR performs much better, since it is able to put more predictors to work. We therefore reiterate the need for adaptive algorithms that will not only learn the nuances of each server, but will also adapt over time as user behavior changes.

The random replacement strategy is the worst performer. This indirectly indicates that Web re-accesses are not random. It is also interesting that FIFO (used by Netscape and Microsoft client browsers) is among the worst performers. Other researchers share similar findings.[6]

FUTURE WORK

Some of the parameters used in this article, such as window size and length of learning, were arbitrarily chosen. We need to perform more experiments to determine optimal window sizes for W_F and W_B. One possibility is to base window size on available cache size. This will allow us to learn from objects that are still in cache. It will also allow a fairer comparison with other algorithms that use knowledge of objects currently in cache.

We model re-access as a binary variable. A better model would take into account the actual number of re-accesses. We can also improve our model by using a continuous learning-prediction model (Figure 2b). Furthermore, we hope to include other predictors in future models. These may include "time to first click" (which may predict a user's interest in the page), number of links, and type of keywords on a page. Such context-based information will allow us to model access as a function of interest.

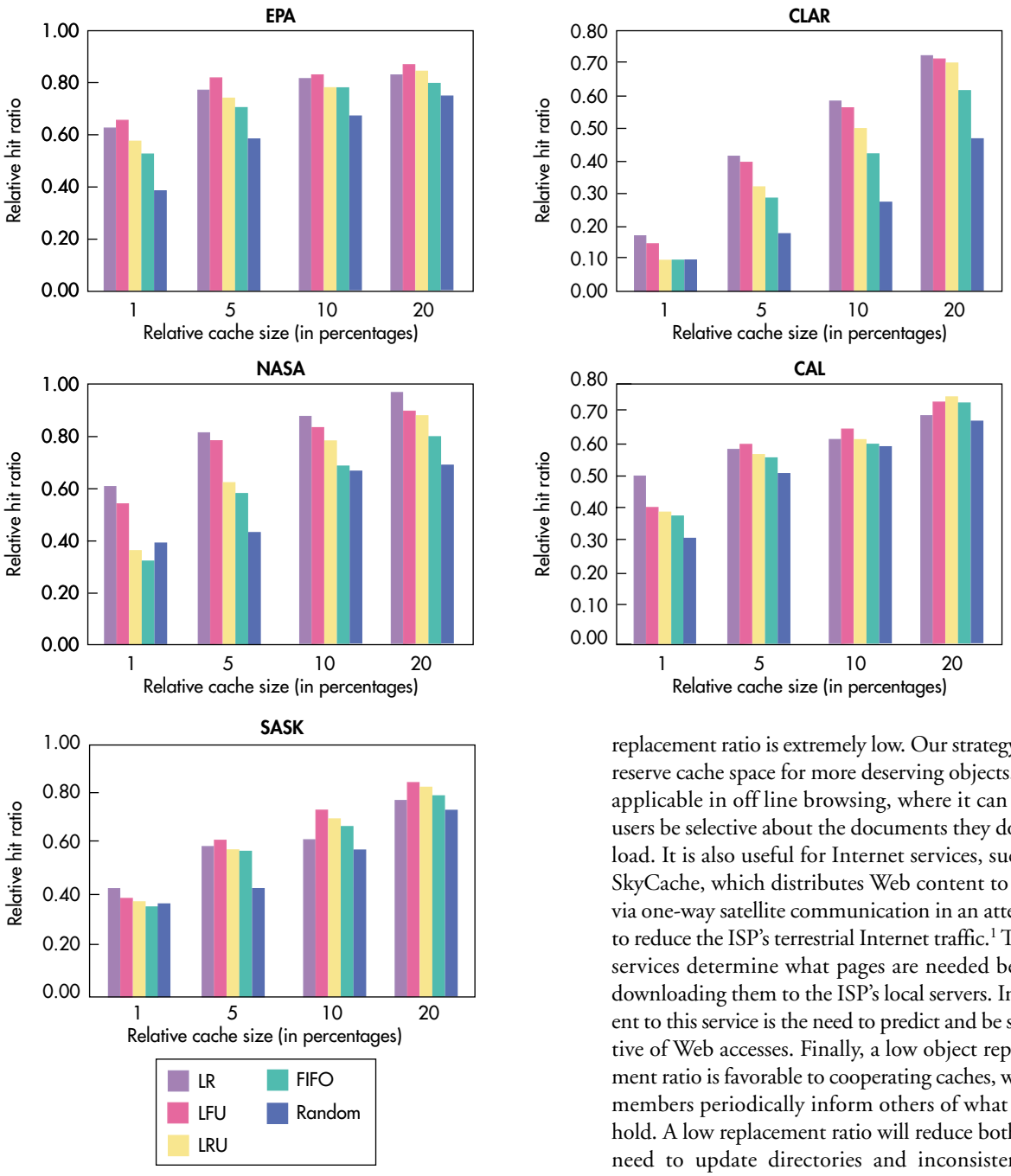

Figure 3. Object hit ratios.

We also hope to test our model on proxy and client caches. Since proxy caches represent a demographically similar group of users, there may be a common pattern in such accesses. An online simulator, using current traces, will also be necessary to obtain realistic cache performance measures.

Due to the selective nature of our strategy, object replacement ratio is extremely low. Our strategy can reserve cache space for more deserving objects. It is applicable in off line browsing, where it can help users be selective about the documents they download. It is also useful for Internet services, such as SkyCache, which distributes Web content to ISPs via one-way satellite communication in an attempt to reduce the ISP's terrestrial Internet traffic.[1] These services determine what pages are needed before downloading them to the ISP's local servers. Inherent to this service is the need to predict and be selective of Web accesses. Finally, a low object replacement ratio is favorable to cooperating caches, where members periodically inform others of what they hold. A low replacement ratio will reduce both the need to update directories and inconsistencies between the directories and actual cache content.

Despite a relatively simplistic prototype, our results are very promising and bode well for harnessing adaptive models to improve cache performance for the WWW. ■

ACKNOWLEDGMENTS

We would like to thank the researchers who collected the Web traces used in this study: L. Bottomley, Duke University; R. Fridman, University of Calgary; S. Balbach, ClarkNet; J. Dumoulin, Kennedy Space Center; and E. Fogel, University of Saskatchewan.

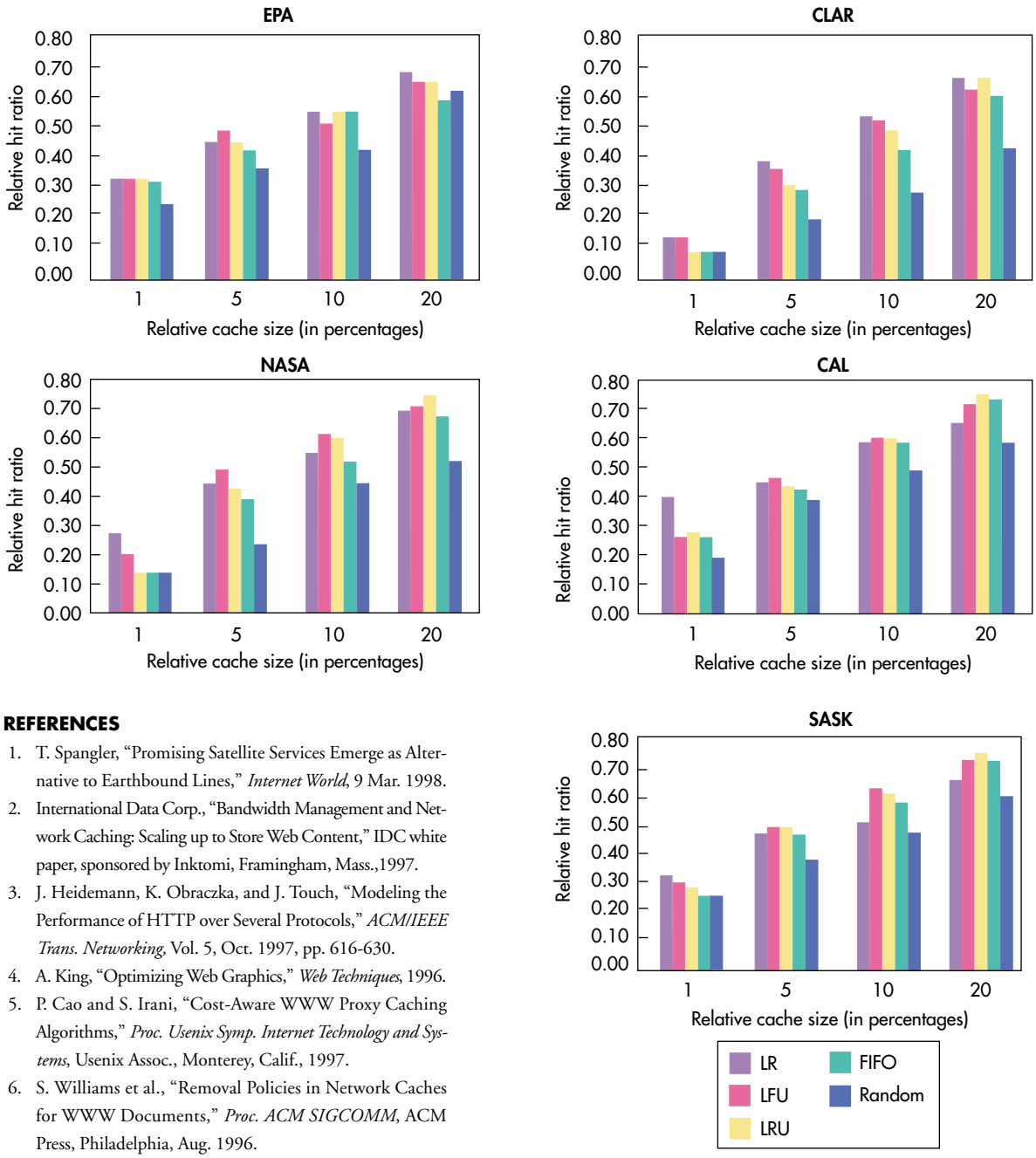

Figure 4. Byte hit ratios.

REFERENCES

1. T. Spangler, "Promising Satellite Services Emerge as Alternative to Earthbound Lines," *Internet World*, 9 Mar. 1998.
2. International Data Corp., "Bandwidth Management and Network Caching: Scaling up to Store Web Content," IDC white paper, sponsored by Inktomi, Framingham, Mass.,1997.
3. J. Heidemann, K. Obraczka, and J. Touch, "Modeling the Performance of HTTP over Several Protocols," *ACM/IEEE Trans. Networking*, Vol. 5, Oct. 1997, pp. 616-630.
4. A. King, "Optimizing Web Graphics," *Web Techniques*, 1996.
5. P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," *Proc. Usenix Symp. Internet Technology and Systems*, Usenix Assoc., Monterey, Calif., 1997.
6. S. Williams et al., "Removal Policies in Network Caches for WWW Documents," *Proc. ACM SIGCOMM*, ACM Press, Philadelphia, Aug. 1996.
7. M. Abrams et al., "Caching Proxies: Limitations and Potentials," Univ. of Virginia Tech. Report TR-95-12, 1995.
8. S. Hosseini, "Investigation of Generalized Caching," PhD thesis, Washington Univ., 1997.
9. P. Lorenzetti, L. Rizzo, and L. Vicisano, "Replacement Policies for a Proxy Cache," 1997; available at http://www.iet.unipi.it/~luigi/caching.ps.gz.
10. V. Almeida and A. de Oliveira, "On the Fractal Nature of WWW and its Application to Cache Modeling," Boston Univ. Tech. Report 96-004, 1996.
11. M.F. Arlitt and C.L. Williamson, "Web Server Workload Characterization: the Search for Invariants," *Proc. ACM SIGMETRICS*, ACM Press, Philadelphia, 1996.
12. M.E. Crovella and A. Bestavros, "Explaining World Wide Web Traffic Self-Similarity," Boston Univ. Tech. Report TR-95-015, 1995.
13. D.G. Kleinbaum, L.L. Kupper, and H. Morgenstern, *Epidemiologic Research: Principles and Quantitative Methods*, Van Nostrand Reinhold, New York, 1982.

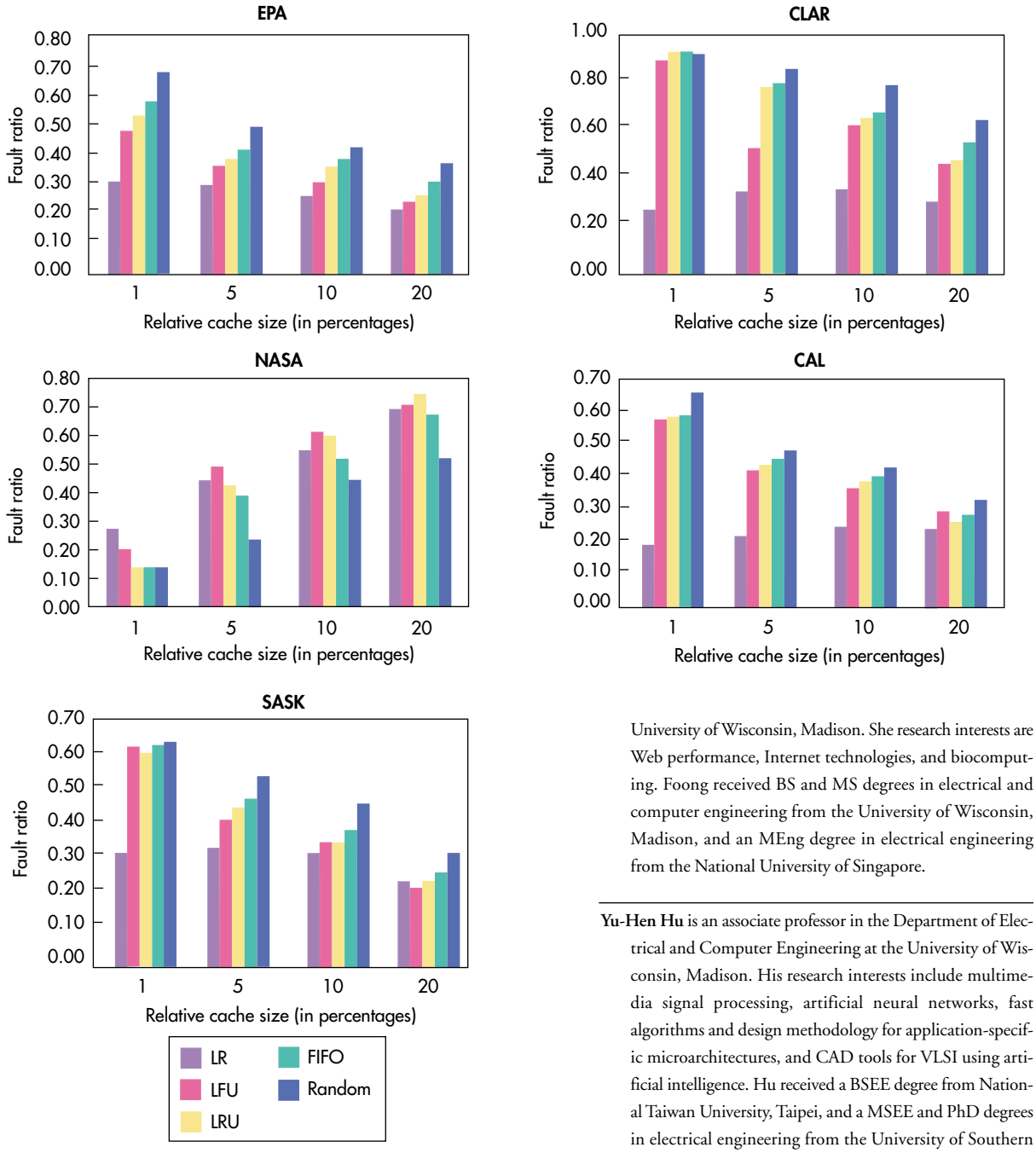

Figure 5. Object replacement ratios.

14. H.S. Stone, *High-Performance Computer Architecture*, Addison-Wesley, Reading, Mass., 1993.
15. Internet Traffic Archive, available at http://ita.ee.lbl.gov/html/traces.html.

Annie P. Foong is an associate information processing consultant in the Department of Surgery and a PhD student in the Department of Electrical and Computer Engineering at the University of Wisconsin, Madison. She research interests are Web performance, Internet technologies, and biocomputing. Foong received BS and MS degrees in electrical and computer engineering from the University of Wisconsin, Madison, and an MEng degree in electrical engineering from the National University of Singapore.

Yu-Hen Hu is an associate professor in the Department of Electrical and Computer Engineering at the University of Wisconsin, Madison. His research interests include multimedia signal processing, artificial neural networks, fast algorithms and design methodology for application-specific microarchitectures, and CAD tools for VLSI using artificial intelligence. Hu received a BSEE degree from National Taiwan University, Taipei, and a MSEE and PhD degrees in electrical engineering from the University of Southern California. Hu is a Fellow of the IEEE.

Dennis M. Heisey is a consultant and associate scientist at the Department of Surgery, University of Wisconsin, Madison. His primary research interest is survival analysis. Heisey received a BS in biology from Pennsylvania State University, an MS in zoology from the University of Michigan, an MS in statistics from the University of Minnesota, and a PhD in biometry from the University of Wisconsin, Madison.

Readers can contact Foong at foong@surgadmin.surgery.wisc.edu